**IceBreaker Execution Plan**

**1. Objective**

Launch IceBreaker’s real-time, ephemeral, proximity-based social connection platform with minimal, scalable, modular architecture — balancing speed, flexibility, and user experience.

**2. Core Development Phases**

| **Phase** | **Goal** | **Outcome** |
| --- | --- | --- |
| Phase 1 | Foundation Setup | Repositories created, Supabase environment ready, Vercel hosting live, base configurations complete |
| Phase 2 | Core Feature Build | Authentication, Interest Management, Proximity Detection, Ephemeral Chat |
| Phase 3 | Privacy, Safety, and Monetization | Visibility toggles, Panic Button, RevenueCat subscription setup |
| Phase 4 | Alpha Testing | Internal usage to validate flows and performance |
| Phase 5 | Beta Testing | Invite-only user testing to collect real-world engagement feedback |
| Phase 6 | Iteration & Launch Prep | Refine UX, fix critical bugs, finalize scalable systems |

**3. Step-by-Step Execution Tasks**

**Phase 1: Foundation Setup**

* Create the GitHub repo following recommended folder structure.
* Set up Supabase project (Auth, Database, Realtime).
* Deploy base Next.js app to Vercel.
* Initialize CI/CD via GitHub Actions (build and deploy triggers).
* Configure environment variables for dev, staging, and production.

**Phase 2: Core Feature Build**

* Implement Supabase Authentication (Google, Apple, Email login).
* Build User Profile Setup with random usernames and interest selection.
* Develop Proximity Matching Engine using Turf.js + Supabase Edge Functions.
* Create Main Interaction Screen (live user cards with proximity scan).
* Build Ephemeral Chat Interface with real-time message expiration.

**Phase 3: Privacy, Safety, and Monetization**

* Add Visibility Toggle for user discoverability control.
* Integrate Panic Button to send real-time location to emergency contacts.
* Connect RevenueCat to manage subscription options and in-app boosts.

**Phase 4: Alpha Testing**

* Recruit a very small trusted user group (internal or close contacts).
* Validate:
  + Signup and onboarding flow.
  + Matching and ephemeral chat accuracy.
  + Panic Button functionality.
* Capture bugs, UX friction, system performance.

**Phase 5: Beta Testing**

* Launch private beta with 100–200 real users.
* Monitor key metrics:
  + Match Rate
  + Chat Initiation Rate
  + Session Duration
  + Privacy Satisfaction
* Collect structured feedback via surveys and bug reports.

**Phase 6: Iteration & Launch Prep**

* Prioritize fixes and improvements based on Beta feedback.
* Tighten UI/UX (polish animations, loading states, feedback prompts).
* Stress-test systems for load (simulated multi-user environments).
* Prepare marketing materials if public launch is imminent.
* Finalize Data Protection & Privacy Statements for app stores (optional).

**4. High-Level Timeline (90 Days)**

| **Weeks** | **Focus** |
| --- | --- |
| 1–2 | Foundation setup and repo config |
| 3–6 | Core feature build (Auth, Matching, Chat) |
| 7–9 | Privacy features, Monetization integration |
| 10–12 | Alpha testing and critical fixes |
| 13–14 | Beta launch and feedback analysis |
| 15–16 | Post-beta iteration and prep for full-scale deployment |

**5. Key Execution Principles**

* **Modularity First:** Build features independently; prioritize hot-swap ability.
* **Move Fast, Protect Core:** It's acceptable to hack features fast, but never compromise user data security.
* **Vibe-Driven Sprints:** Intuitive "flow coding" is good — but checkpoint against clear module completion weekly.
* **No Premature Scaling:** Optimize stability for hundreds, not millions, in the first 90 days.
* **Feedback-Led Growth:** Let user feedback heavily influence the first 5 major feature tweaks after Beta.